2022

Learning Robot Motion from Creative Human Demonstration

Charles C. Dietzel
Virginia Commonwealth University

VCU College of Engineering
Electrical and Computer Engineering

Learning Robot Motion from Creative Human Demonstration

Virginia Commonwealth University
College of Engineering
Department of Electrical and Computer Engineering
601 West Main Street
Richmond, Virginia 23284–3068

Learning Robot Motion from Creative Human Demonstration

A thesis submitted in partial fulfillment of the requirement for the degree of Master of Science at Virginia Commonwealth University.

by

Charles Dietzel
Bachelor of Science in Electrical Engineering, Virginia Commonwealth University, 2021

Director: Patrick Martin
Assistant Professor, Department of Electrical and Computer Engineering

Virginia Commonwealth University
Richmond, Virginia
December, 2022

Acknowledgement

I would like to thank my parents for supporting me over the years and for teaching me the value of trying my best. I would also like to thank Dr. Martin for helping me find my way through this wild ride called research. I would also like to thank my thesis committee, who have graciously given me their time and attention over the past few years. Finally, I would like to thank all of my fellow research students that I have had the pleasure of working with over these past few years: I have learned so much from all of you.

Contents

List of Figures

List of Tables

Abstract

Learning Robot Motion from Creative Human Demonstration

By Charles Dietzel

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

Virginia Commonwealth University, 2022.

Major Director: Patrick Martin, Assistant Professor, Department of Electrical and Computer Engineering

This thesis presents a learning from demonstration framework that enables a robot to learn and perform creative motions from human demonstrations in real-time. In order to satisfy all of the functional requirements for the framework, the developed technique is comprised of two modular components, which integrate together to provide the desired functionality. The first component, called Dancing from Demonstration (DfD), is a kinesthetic learning from demonstration technique. This technique is capable of playing back newly learned motions in real-time, as well as combining multiple learned motions together in a configurable way, either to reduce trajectory error or to generate entirely novel motions based on user specified parameters. DfD was utilized to enable a cooperative robot-human dance performance, and that performance has been evaluated to demonstrate that DfD achieved its design goals. The second component of this newly developed robot control framework, called Pose Energy Correspondence Mapping (PECM), is a passive-observation based learning from demonstration technique which is used to convert human pose trajectories into corresponding robot joint trajectories. This conversion process makes use of an energy based neural network model in order to attempt to achieve high quality results with a minimally sized training set. These two components have been combined together and the resulting framework has been evaluated by means of a comparative survey between human-generated robot motions, PECM-generated robot motions, and robot motions generated by a baseline neural network technique. These survey results are analyzed and discussed in order to identify the strengths and limitations of the newly developed framework.

Chapter 1

Introduction and Background

1.1 Introduction

In the near future, robots of various shapes and sizes will become increasingly intertwined in people's day-to-day lives in a variety of environments, such as in factories [1], highways [2], and healthcare [3]. This increasing interdependence among humans and robots has revealed a few significant shortcomings of many robot control frameworks in use today.

One major limitation of modern robots is the difficulty required to "teach" these robots new skills. Previous generations of robots, such as those commonly used in factory assembly lines, are limited to structured environments and predictable behaviors, with little ability to generalize in novel or unexpected situations [1]. These robots are only capable of repeating the trajectories and behaviors that they were manually programmed with by a human expert. This form of robot control is effective in a heavily regulated environment alongside an expert with the requisite knowledge to program and update the robot with new motions.

More recently, there has been a desire for robots that operate alongside humans in less constrained conditions, without requiring expert intervention. One promising area of modern research to resolve this problem is known as Learning from Demonstration (LfD) [3]. The idea behind LfD is that instead of robots needing to be manually and explicitly programmed

with new behaviors by an expert, the robot learns new motions and behaviors directly from humans in the environment by "watching" a human perform a demonstration of the motion to be learned.

Even given that a robot can learn from human demonstrations, there are still many areas in which current robot control systems struggle. For instance, many robots controlled by LfD systems are unable to replay newly learned motions right away, requiring some computationally intensive and time consuming training process first [4]. Ideally, LfD techniques would be designed such that they could reproduce newly demonstrated motions in real time. In this way, it would be possible for the human performing the demonstration to get instant feedback on how the demonstration was progressing and how the motion they intended to demonstrate was being learned by the LfD algorithm.

Another useful feature which relatively few LfD methods possess is the ability to reduce demonstration error using repeated demonstrations [5]. Due to human nature, it is inevitable that when a demonstration is performed, there will be some minor mistakes or variations. One way an LfD system could handle this problem would be to allow human demonstrators to provide multiple demonstrations of each motion to be learned. The LfD system could then perform some operation on this set of demonstrations to merge them into one "canonical" demonstration with minimized error.

One other feature that is useful for an LfD system is the ability to creatively combine multiple demonstrations [6]. The idea is that, after the robot is trained with a variety of demonstrated motions to perform some task or tasks, a user may select combinations of these motions to merge. The user would also be given the option to adjust parameters that define exactly how this combination process would take place. The newly generated motion resulting from this process would contain more complex and diverse motions than what the robot originally learned, customized by the user specifically to perform a new motion without requiring a new demonstration. One possible use for this sort of creative motion generation would be in gesture-based human-robot interaction [7, 8], where motions could be generated

as needed to convey unique meanings from the robot to humans.

This thesis presents a novel robot control framework that satisfies the previously described features and criteria. This control framework is split into two components, each of which handles a different aspect of the robot's operations. One component, known as Pose Energy Correspondence Mapping (PECM), is a passive observation LfD technique which is designed to solve the correspondence problem and to allow human demonstrations to be learned by the robot quickly and effectively. The other component, known as Dancing from Demonstration (DfD), is a robot control framework capable of playing back newly demonstrated motions in real-time, using multiple demonstrations of the same motion to remove noise and errors, and merge multiple different demonstrations to generate an entirely novel motion. These two components are combined together and implemented on a real robot, and the quality of the resulting control system is evaluated.

1.2 Background

1.2.1 Learning from Demonstration

Within LfD, there are three overarching methods by which tasks can be demonstrated to a robot. The first of these is kinesthetic teaching, where a human physically moves a robot manipulator through the motion being demonstrated [3]. This technique is convenient since the motion to be learned is already provided in the robot's own joint space. As such, it is not necessary to map each learned motion from the human body to the different kinematics of the robot. Also, this category of methods only requires the joint position sensors that are built into the robot's actuators, thereby reducing the total cost and complexity of the imitation learning system. However, humans cannot easily kinesthetically manipulate high-DoF robots, heavy robots, or those robots with many different limbs that must move in coordinated ways, such as multi legged robots. This issue makes kinesthetic teaching infeasible in those cases. Due to these limitations, recent research has been dedicated to developing task

demonstration methods that do not suffer from these issues.

Another LfD technique that alleviates some of these issues is the use of teleoperated demonstrations [3]. This technique involves a human using a controller or remote input device to manipulate the robot. Motion demonstrations are performed by using the controller to make the robot perform an action. However, for teleoperation to work, a controller interface must first be designed, which may be difficult or undesirable in many environments. Additionally, the control device may not be intuitive or precise enough to allow the human performing the demonstration to accurately reproduce the desired task behavior using the robot. Due to the limitations of both of these approaches, for some imitation learning tasks, neither teleoperation nor kinesthetic teaching are viable methods for task demonstration.

Research has also been done in developing task demonstration techniques using the idea of passive observation [3]. Observation techniques use external sensors to measure and record the pose and/or action of the human motion demonstrator directly. The captured human motion data is mapped to a motion primitive or trajectory that the robot can use. Observation LfD has a number of advantages, such as the fact that it can be applied to robots of arbitrary size or complexity, an advantage over kinesthetic learning. It also requires no external control mechanism to be developed, unlike teleoperated methods. It is also intuitive for the human task demonstrator, only requiring them to perform a task with their own body while their motion is being recorded by external sensors. However, as previously mentioned, to make use of these recorded human motions, there must be some method of mapping them onto a joint or trajectory space that the robot's control system can utilize. As an additional challenge, the robot and human action spaces often have significantly different degrees of freedom. Solving this task of mapping an arbitrary human motion to a corresponding robot motion is known as the correspondence problem [9].

Given these respective sets of strengths and limitations, it was decided to use observation LfD for the PECM technique developed in this paper. It requires no controller, unlike teleoperated approaches, and much more feasible and intuitive on complex robots than kinematic

approaches. However, due to the selection of observation LfD, the correspondence problem needs to be solved.

1.2.2 Solutions to the Correspondence Problem

The correspondence problem, which is the task of mapping human poses to some robot action space is analogous to generating a policy in reinforcement learning (RL). In an RL context, a policy is some mapping between a given system state and an action that corresponds to that state. In this context, the recorded human pose is the system state and the corresponding desired robot joint positions are the corresponding action. Therefore, to effectively implement observation-based LfD, it is necessary to generate a policy to perform the task of mapping the recorded human pose states to corresponding robot actions [2]. To do this, a variety of different approaches were considered.

Reinforcement Learning

One approach which immediately seems intuitive is to make use of reinforcement learning. RL techniques such as Generative Adversarial Imitation Learning (GAIL) or Adversarial Inverse Reinforcement Learning (ARIL) are designed to generate policies based on expert demonstrations [10, 11]. By using these techniques, one could theoretically develop a policy mapping human poses to robot joint space. However, essentially all RL techniques require online training to generate a policy, meaning that they require the RL agent to explore its environment. In practice, training an effective RL agent requires a computer simulation of the environment, which can be explored significantly more quickly, cheaply, and safely than the real world. In order to build a simulated environment that could algorithmically determine the degree of correspondence between a human pose and a set of robot joint angles, it would by definition require solving the correspondence problem. Therefore, since no simulation environment of the correspondence problem can be built without first solving the correspondence problem, RL cannot be used to train an agent to solve the correspondence

problem.

Behavioral Cloning

One alternative to reinforcement learning which does not need online training is known as behavioral cloning [12]. Behavioral cloning is the name for a category of supervised learning techniques which are designed to learn a policy exclusively using labeled demonstration data. These techniques are not reinforcement learning based, and as such do not require a simulation environment, meaning that they can be applied to practically any type of problem. There are a few theoretical downsides to behavioral cloning techniques. First, since the only information they have about the problem space is contained in the labeled demonstration data, behavioral cloning techniques do not inherently generalize well beyond the context in which they were trained [3]. Also, behavioral cloning assumes that each state action pair they are trained on is i.i.d, which is not necessarily the case. When the i.i.d assumption is violated, it can cause a compounding error problem where a small amount of error in one predicted action can send the system to a configuration that the policy was not trained on. Eventually, the system ends up in a state where the desired action is undefined by the training data, resulting in completely nonsensical behaviors. The easiest way to compensate for these issues is to provide more training data, which is one of the main reasons why behavioral cloning techniques require more demonstration examples than comparable reinforcement learning techniques. On the other hand, reinforcement learning techniques are less sample efficient than behavioral cloning techniques since they must sample additional information about the system's behavior from the environment during training. This environment sampling process is computationally intensive during training.

The current state-of-the-art (SOTA) behavioral cloning technique is known as Implicit Behavioral Cloning (IBC), provided in [12], which describes the use of Energy-Based Models (EBM) instead of traditional supervised learning models, which use Mean Square Error as a metric. Because explicit models define a function, they are unable to directly represent

discontinuities in a problem space. They are also unable to represent set-valued functions. EBM models define their input-output mapping implicitly, so they can represent a wider space of possible policies more effectively. Since EBM models implicitly learn the "shape" of the input-output mapping, they tend to generalize better than explicit models where the problem space has little or no training data available. More details on the theoretical advantages of EBMs in imitation learning are found in [13]. These properties significantly improve the performance of IBC on many robotics tasks and benchmarks.

Offline Reinforcement Learning

A variant of reinforcement learning which does not require environment access is known as offline reinforcement learning, also known as full batch reinforcement learning [14]. Offline reinforcement learning can be thought of as a generalization of behavioral cloning with fewer constraints. Like behavioral cloning, offline reinforcement learning agents learn exclusively from a static dataset of previously obtained environment accesses. However, unlike in behavioral cloning, offline reinforcement learning datasets contain reward information associated with specific actions, instead of providing the optimal action for each state. However, offline reinforcement learning datasets do not always have any reward information present for certain actions within demonstrations. Additionally, offline reinforcement learning agents can take into account multiple states and actions from the past, unlike behavioral cloning techniques which make predictions only from the state at the current timestep. These properties make offline reinforcement learning techniques ideal for agent control problems without the possibility of environment access. However, offline reinforcement learning is a recently established field, being introduced for the first time in 2019 [15]. Additionally, training an agent to perform a policy given only a fixed dataset of off-policy demonstrations with potentially sparse labels is a difficult task. As a result, it is conceivable that behavioral cloning techniques may perform better on certain tasks than SOTA offline reinforcement learning methods, given that behavioral cloning is a much older and more well studied problem. Also,

the added complexity of implementation required by offline reinforcement learning methods may be undesirable for some tasks where behavioral cloning performs well by itself. Finally, in general, reinforcement learning techniques train more slowly than comparable behavioral cloning models, which may be a significant disadvantage in some cases.

There are a few offline reinforcement learning methods that are all competitive in various benchmarks for SOTA. Each of these techniques are based on Q-learning, which is a technique originally designed for online reinforcement learning. One problem with this is that offline reinforcement learning involves training the RL agent using datasets that significantly differ from the previously learned policy, something which never happens in online reinforcement learning[15][16]. This discrepancy between the off-policy training data and the newly learned behavior policy results in something called the "distribution shift" problem, where the learned Q-function fails to accurately extrapolate the value of a given training sample which is far from the distribution of previously learned data. In general, the extrapolation error present in these cases is an overestimate of the true value of a given training sample. This overestimation can result in the Q-function becoming disproportionately biased towards out-of-distribution actions, which generally significantly harms the results of the learned policy at test time. The core innovation of each of these offline reinforcement learning methods is some unique method of solving the distribution shift problem by simultaneously restricting the allowed extrapolation error from the Q-function while also allowing some reasonable extrapolation to take place, so that the Q-function will eventually learn an optimal policy given all the data in the training set.

Technique Comparison

To determine which of these approaches to use in solving the correspondence problem, current SOTA techniques were collected and the theoretical advantages and justifications for each technique were considered. Additionally, each of the previously described behavioral cloning and offline reinforcement learning techniques were tested on the Adroit and FrankaKitchen

task sets of the D4RL benchmark suite [17]. The results of these analyses are presented in Table 1.1.

Table 1.1: Comparison of Potential Techniques for Solving the Correspondence Problem

Technique Name:	**Technique Type:**	**Technique Description:**	**Theoretical Advantages:**	**Benchmark Results:**
Implicit Behavioral Cloning (IBC) [12]	Behavioral Cloning	Supervised learning approach formulated as a conditional energy-based modeling problem	Can represent discontinuities and multi-valued functions with arbitrary precision, extrapolates well outside training data	Overall best results of all tested techniques. Tends to do especially well on datasets of high-quality human demonstrations.
Conservative Q-Learning (CQL) [16]	Offline Reinforcement Learning	Applies a regularization term to its Q-function in order to learn a conservative estimate of each true Q-value	Conservative Q-value estimates for out-of-distribution samples limit distribution shift problem, simple to implement.	Overall good results, roughly equivalent to PLAS. Tends not to do especially well or poorly at any task.
Implicit Q-learning (IQL) [18]	Offline Reinforcement Learning	Instead of updating the Q-function directly, an approximation is used which implicitly finds the optimal in-distribution action for a given state	Easy to implement and computationally efficient. Results improve with online fine-tuning after being initialized offline.	Overall excellent results, better than PLAS/CQL but worse than IBC. Tends to do especially well on easy tasks.
Policy in the Latent Action Space (PLAS) [19]	Offline Reinforcement Learning	Models the learned policy in the latent action space of a Conditional Variational Autoencoder, implicitly constraining generated actions within the dataset distribution	Allows generalization within the dataset. Can be configured to allow a controlled amount of out-of-distribution actions if desired.	Overall good results, roughly equivalent to CQL. Tends to do especially well on hard tasks and poorly on easy tasks.

IBC [12] tended to perform the best when full human demonstration datasets were available. By contrast, Conservative Q-Learning (CQL) [16], Implicit Q-learning (IQL) [18], and Policy in the Latent Action Space (PLAS) [19] all traded positions as the best performer when the dataset contained extraneous actions (not representing the desired policy). Overall, IBC had the best performance over the widest range of tasks compared to the other

techniques.

When evaluating which of these techniques is most optimal to use in solving the correspondence problem, the exact structure of the correspondence problem should be considered. Specifically, the correspondence problem involves mapping some set of human poses trajectories to some set of robot joint trajectories. Since this dataset has been collected in our lab under controlled conditions, it was possible to ensure that all the human/robot demonstrations were collected with no extraneous actions present. IBC performed best overall on the D4RL benchmarks, and it performed especially well when tested on tasks with no extraneous actions in the dataset [12]. IBC also has a number of theoretical advantages which make it very compelling for use in robotics tasks. The combination of these advantages made IBC the clear choice to select for solving the correspondence problem within our PECM technique.

1.2.3 Playback of Learned Motions

Using PECM or some other LfD technique, it is possible for the robot to generate joint trajectories based on some demonstrated human motion. Given these joint trajectories, it was also necessary to consider the best technique to playback these learned motions, while satisfying the other desired criteria of the newly developed robot control system. As stated in Section 1, the desired control system should be able to playback newly learned motions in real time, use repeated demonstrations of the same motion to reduce error, and generate novel motions using combinations of previously learned motions and motion parameters provided by a human user. One important prior LfD technique, presented in [20], known as Learning from Demonstration by Averaging Trajectories (LAT), already possessed many of these desired attributes. LAT has a number of design limitations which render it unsuitable for use here without modifications and improvements. However, given LAT's flexible design, it seemed possible to modify LAT in order to create DfD, an algorithm that would satisfy all the desired criteria, and could be integrated successfully with PECM.

Chapter 2

Dancing From Demonstration

The first half of the robot control framework developed in this thesis is known as Dancing from Demonstration (DfD), which was recently published in [6]. There are a wide variety of techniques that can be used to enable a robot to learn motion data from human demonstrations. Once these robot motions are generated, some other technique is required to enable the robot to perform these motions using its manipulator. DfD is a kinesthetic learning from demonstration (LfD) technique capable of performing both of these functions. DfD was initially conceived as a choreography framework to enable a robot-human duet dance performance, as described in [6]. This original task requirement was the motivating factor behind all of the design requirements and capabilities that DfD was developed with.

One of the primary design goals of DfD was to design an LfD system capable of learning, processing, and replaying robot motions in real-time. In order to satisfy this requirement, DfD was designed with a focus on simplicity and efficiency. Many other LfD systems, such the Gaussian Process based approach described in [4], require several seconds to process a single 10 second learned motion. This motion processing delay renders these techniques sub-optimal for use cases that involve the learning and immediate playback of new motions. By contrast, DfD is capable of processing a similar 10 second learned motion in under a second. The benefit of this real-time operation capability is that it enables the robot to participate

in tasks where a robot and a human interact and cooperate simultaneously, including the dance in [6].

It was also necessary for DfD to have the ability to learn new motions with only a very small number of demonstration samples, or potentially just one. This also enabled its use in real-time settings where time available to perform a motion demonstration was very limited. Additionally, it was desirable for DfD to only require data from the robot's proprioceptive sensors in order to function. This way, DfD would be flexible even in environments where external sensors were difficult or impossible to set up. These features were also necessary to enable tasks such as duet in [6].

Additionally, DfD has the goal of enabling the robot to "modulate" multiple similar motions, combining them together to create an entirely unique and novel motion. When designing the motion modulation capability of DfD, there were two primary use-cases in mind. One desired feature was for the modulator to be able to find the "average" of two or more similar motions, as a means of smoothing out errors in the recording of any particular motion. This way, if a human teacher demonstrated the same motion multiple times, the various recordings of that motion could be combined into one refined version with minimal recording errors or imperfections. The second desired feature for DfD was some method by which multiple different recorded motions could be combined together to generate an entirely new motion, "inspired" by the originals. Additionally, by adjusting the parameters of the combination process, the exact degree to which each original motion influenced the generated motion could be adjusted. This feature is primarily useful in creative contexts, such as the dance performance in [6], where the robot is told to perform a new motion based on some set of motions it already knows. By adjusting the modulation parameters, a variety of new motions can be generated based on the same set of original motion recordings. This enables a dance choreographer to easily modify the robot's dance routine without needing to record entirely new motions.

2.1 Dancing from Demonstration Algorithm

In order to develop all of these capabilities for DfD, a technique known as Learning from Demonstration by Averaging Trajectories (LAT) [20] was used as a basis. LAT came close to satisfying many of the qualities desired in DfD, and was designed in a flexible way that was easy to adapt and improve to achieve the desired functionality. LAT was originally designed as a robot control system for performing manipulator motions that physically interacted with an object from the environment.

To make use of LAT, the robot user first kinesthetically manipulates the robot arm to demonstrate a desired motion which performs some task involving moving an environment object. This process can be repeated a desired number of times for each motion. Once the user is satisfied with the performed motion demonstrations, the LAT algorithm then uses all demonstrations of that motion to learn one finalized motion, constraining that finalized motion depending on the position of the environment object throughout the demonstrations. Because of this design choice, LAT required that each motion demonstration have at least one interaction with some object external to the robot, and the position of that object relative to the robot needed to be tracked with sensors throughout the motion. This was undesirable behavior for DfD, which needed to be able to reproduce motions without any object interactions and without the use of external sensors that could be used to track the position of external objects. Additionally, while LAT was not very computationally complex, it was still possible to make efficiency improvements by redesigning its underlying algorithm.

One algorithm fundamental to the functionality of LAT is known as Dynamic Time Warping (DTW) [21], which is a method of time-aligning two similar signals that have some unknown time offset from each other. DTW has a few shortcomings, such as the fact that the alignment quality degrades rapidly as the signals to be aligned become more dissimilar, such as with the addition of noise. Additionally, DTW can only perform alignment of two signals at once, meaning that three or more signals must be aligned through some other approach. The most effective approach to do this involves DTW Barycenter Averaging (DBA) [22],

which generates even worse results than regular DTW, since it involves iteratively applying DTW to each signal until convergence is reached. Thus, any signal artifacts that would have been generated by DTW are equally bad or worse when using DBA, because error from each iteration accumulates. Additionally, LAT does not use DBA, instead choosing to use an approach where every possible pair of signals is DTW aligned, and the alignments are kept only for the pairs including the particular signal with the lowest total warping distance. This alternative method of aligning more than two signals has worse efficiency and time-complexity than DBA, while still resulting in significant error.

One major improvement that was made to LAT during the design process of DfD was to replace DTW [22] with Soft-DTW [23]. Soft-DTW is a variation of DTW that uses a soft-min operator instead of the min operator used in DTW. This has the effect of greatly improving alignment results for noisy signals, without significantly increasing computational complexity. Additionally, while the limitation of only being able to align two signals still exists, Soft-DBA generates results that are significantly better than those of DBA, and so Soft-DBA has been used to replace the multi-signal DTW technique used by LAT.

Algorithm Design and Implementation

The DFD algorithm [6], enumerated in Algorithm 1, operates on robot joint trajectories and generates new robot motions with desired modulation parameters. Figure 2.1 illustrates the process for the `WaveLeft` example gesture that was developed for the performance in [6]. More formally, assume that a human demonstrates a finite number of motions that track with an index, $s \in \mathcal{S} \subset \mathbb{N}$. Each demonstration is captured as a sequence of joint angle vectors, $\theta \in \mathbb{R}^n$, where n is the number of robot manipulator joints, at some set sample rate, r, for a time duration, d_s. Therefore, we define a recorded demonstration as $D_s = \{(k, \theta(k)) | k = 0, ..., rd_s\}$. The left plot of Figure 2.1 shows three demonstrations sampled at 50Hz, each with their own value for d_s.

The user supplies the set of demonstrations, $\{D_s\}$, $m \leq s$ desired demonstrations, and

Algorithm 1 Dancing from Demonstration algorithm

Require: Set of demonstrations, $\{D_s\}$, number of motions to modulate, m, weight vector, W

$M \leftarrow \{(D_j, W)|j = 1, ..., m\}$

$D_{avg} \leftarrow$ `soft-dtw-barycenter`(M)

$l_w \leftarrow \sum_{i=1}^{m} w_i|D_i|$

$D_w \leftarrow$ `resample`(D_{avg}, l_w)

$M_{aligned} \leftarrow$ `soft-dtw-align`(M, D_w)

$D_{modulated} \leftarrow$ `weighted-avg`$(D_{aligned}, W)$

return $D_{modulated}$

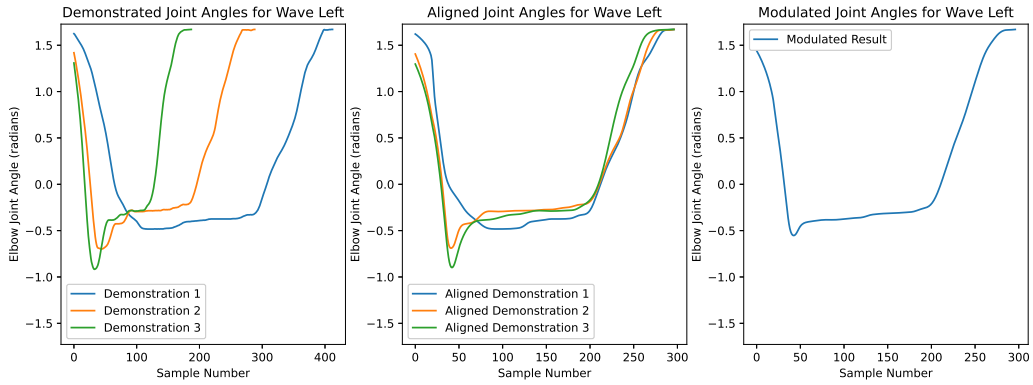


Figure 2.1: This figure demonstrates how the DfD algorithm operates on the elbow joint angle's demonstrated trajectory as part of our `WaveLeft` example gesture. The plot on the left shows the captured demonstration set with three motions, e.g. $s = 3$. The middle plot shows the output of the Soft-DTW alignment process when we set $m = s$ and use a balanced weight vector of $W = [\frac{1}{3}\ \frac{1}{3}\ \frac{1}{3}]$. The resulting trajectories have the same start and end points based on the value of l_w. The plot on the right shows the final modulated motion using the weight vector.

a vector of m modulation weights, $W \in \mathbb{R}^m$. We define this modulation set as $M = \{(D_j, W)|j = 1, ..., m\}$, where $\forall w_i \in W, \sum_{i=1}^{m} w_i = 1$. In our example in Figure 2.1, we use a balanced weight vector, $W = [\frac{1}{3}\ \frac{1}{3}\ \frac{1}{3}]$; however, a user may choose any combination that satisfies the weight constraint.

This modulation set is fed into a Soft-DTW barycenter averaging [23] procedure[1] and the modulated set $D_{avg} = \{(k, \theta(k))|k = 0, ..., l\}$ is generated, where $l = \max_{s \in \mathcal{S}} |D_s|$. The weighted length $l_w = \sum_{i=1}^{m} w_i|D_i|$ is computed, and the new trajectory is then resampled to length l_w, which results in the an approximate modulated demonstration, $D_w = \{(k, \theta(k))|k = 0, ..., l_w\}$. Soft-DTW alignment is applied to the original demonstrations in

[1] `https://tslearn.readthedocs.io/en/stable/`

the modulation set, M, and the approximate demonstration, D_w, resulting in a new motion set, $M_{aligned} = \{(D_{aligned,j}, W)|j = 1, ..., m\}$. The middle plot in Figure 2.1 shows this alignment to the weighted average of the demonstration lengths, l_w. Finally, the weighted average of each $D_{aligned,j}$ with its corresponding weight in W results in the final motion, $D_{modulated}$. Based on our example's balanced weighting, one can see in Figure 2.1's right plot that the modulated motion looks like the average of the inputs. This approach allows a user to experiment with a mixture of weights to achieve a desired motion effect from the robot, while also keeping the core characteristics of the initial motion demonstrations.

2.2 Performance Results

Since the primary novel feature of DfD is a motion modulation process designed to enable creative motion generation and dance choreography, we chose to evaluate DfD by performing a robot-human dance as described in [6]. The robot model used in the performance was an Interbotix LoCoBot equipped with a 5 degree-of-freedom (DOF) manipulator arm. The DfD algorithm was implemented in ROS and a state machine framework called FlexBE [24] was used in order to design and modify the performance structure.

The performance venue that we used did not have any spatial localization system for the robot to use, so the robot's position estimation was based entirely on its internal odometry. Since the performance was only about 5 minutes long, odometry error did not have much time to accumulate, so the negative effect of the resulting positional inaccuracy was not large enough to significantly impact the performance. That said, plans are in progress to implement another spatial positioning sensor to reduce the robot's position estimation error in future performance tasks.

The performance[2] is split into three sections. The first minute of the performance happens at the front of the stage, where the human dancer improvises a new motion during the performance and teaches three different samples of it to the robot using DfD, as shown in

[2] `https://vimeo.com/678480077/`

Figure 2.2. After the new motion has been demonstrated in this way, DfD motion modulation is applied in order to smooth out any demonstration error that may have been present. The robot then immediately plays this modulated motion back as shown in Figure 2.3, visually demonstrating that it has successfully learned this new gesture primitive. This motion learning process makes use of most of the capabilities that DfD has been designed with. First, the robot must learn a new motion with a very small amount of training data, only a few demonstration samples. Also, DfD must be able to modulate these multiple motion demonstrations in order to generate one combined motion with reduced demonstration error. Finally, the whole process must run in real-time, so that the robot can immediately play back the motion and continue the performance after modulation is complete.

Figure 2.2: The beginning of section one of the performance, where the human dancer is teaching an improvised motion to the robot

Figure 2.3: The end of section one of the performance, where the robot is playing back the newly learned motion

After section one of the performance is complete, the robot begins stage two, where it plays back a sequence of pre-recorded motion primitives in sync with its human partner while moving across the stage, as shown in Figure 2.4. This phase of the performance makes use of the final functionality built into DfD, the ability to creatively generate new motions as some user specified blend of a set of pre-existing motion demonstrations. More specifically, all of the motions that are played back during this section are the result of the DfD creative

motion modulation process.

Finally, in section three of the performance, the robot and human separate as shown in Figure 2.5, and each performs a dance solo across the stage from one another. After they have both finished their solo, they meet back up at the front of the stage for the end of the performance, where they both perform the same motion in sync once again. Throughout section two and three of the performance, the new motion that the robot learned in part one is played back multiple times in order to add an element of dynamicism.

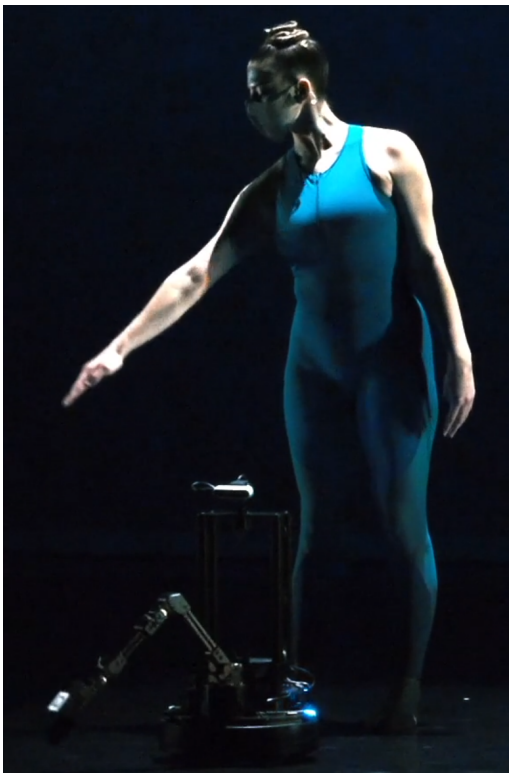


Figure 2.4: Section two of the performance, where the human and robot are performing the same motion in sync

Figure 2.5: Section three of the performance, where the human and robot separate to perform their own dance solos

This robot-human duet was performed multiple times, each time with a different motion being improvised and taught to the robot in phase one. Each time it was performed, this dance performance worked successfully. This demonstrates that DfD is an effective LfD framework for use in real-time, creative motion learning environments where minimal training data is available.

Chapter 3

Pose Energy Correspondence Mapping

The development and use of DfD in the robot/human dance in [6] resulted in the recognition of certain limitations with the approach. Specifically, DfD is a kinesthetic LfD technique, meaning that a human demonstrator must physically grab and move around the robot's manipulator to teach a new motion. This kinesthetic teaching process is slow, clunky, and does not scale well to larger robots and/or robots with manipulators that have higher degrees of freedom. A better LfD method was desired to solve these issues and research was then performed in order to identify the best possible design for this new technique. As described in Section 1.2, observation LfD is the ideal demonstration type for the new method, and IBC [12] was selected as the best technique to solve the observation LfD correspondence problem. These two techniques have been combined in an effort to replace the kinesthetic LfD functionality built into DfD, and the resulting method is called Pose Energy Correspondence Mapping (PECM).

At a high level, the challenge of PECM, like all passive observation based LfD techniques, is in solving the correspondence problem. The correspondence problem is the task of finding some algorithmic method of mapping a set of human motion sensor observations to a cor-

responding set of robot control outputs that represents that same motion. The difficulty is that, in general, many plausible mappings are possible and it is unclear what the ideal mapping is. Any model or algorithm that performs this task is considered a generative model, which takes human pose data as an input and generates a corresponding robot control output. Over the last few years, a great deal of research has been done regarding the numerous advantages of energy-based models (EBMs) when applied in generative contexts. Therefore, PECM makes use of EBMs in order perform to its pose mapping task.

Given human pose observations $\boldsymbol{o}$ and corresponding robot goal action $\boldsymbol{a}$, a classical explicit feed-forward neural network F_θ, parameterized by θ, is structured in the form $\hat{\boldsymbol{a}} = F_\theta(\boldsymbol{o})$. By contrast, an analogous energy-based model E_θ is structured in the form $\hat{\boldsymbol{a}} = \arg\min_{\boldsymbol{a}} E_\theta(\boldsymbol{o}, \boldsymbol{a})$ [12].

EBMs are a subcategory of what are known as contrastive learning models. These models, described in [25], are designed to learn some representation of each input which simultaneously minimizes that input's distance from other similar inputs and maximizes the distance from dissimilar inputs. In other words, these models attempt to learn a function with outputs that correspond to the density of similar data in each part of the dataset. Therefore, these models can be said to learn the data density distribution of the dataset. By learning this data distribution, the model can then be used to generate new robot actions similar to those that correspond to human pose observations similar to those in the dataset.

In order to train and generate prediction samples from such a model, a more complex process is required compared to an explicit neural network. The goal of the training process is to generate the energy function E_θ that defines the density function of some desired model data distribution. In order to do this, a loss function known as InfoNCE [12] is used which works as follows.

Given some dataset of I observations and corresponding actions $\{\boldsymbol{o}_i, \boldsymbol{a}_i\}$, it is first necessary to compute the data bounds $\boldsymbol{a}_{min}$ and $\boldsymbol{a}_{max}$ for each category in $\boldsymbol{a}$ [12]. Next, some batch of $N-1$ action counter-examples $\{\tilde{\boldsymbol{a}}_i^n\}_{n=1}^{N-1}$ is generated for each $\boldsymbol{o}_i$ by random sampling

from the rectangular region constrained by $\boldsymbol{a}_{min}$ and $\boldsymbol{a}_{max}$ [26]. Each set $\boldsymbol{A}_i$ of N samples, comprised of $N-1$ counter-examples and one corresponding positive sample $\boldsymbol{a}_i$, is used to optimize the InfoNCE loss function:

$$\mathcal{L}_{\text{InfoNCE}} = -\sum_{i=1}^{I} \log \left[\frac{e^{-E_\theta(\boldsymbol{o}_i, \boldsymbol{a}_i)}}{\sum_{j=1}^{N} e^{-E_\theta(\boldsymbol{o}_i, \boldsymbol{A}_i^j)}} \right]$$

EBMs provide additional complications at inference time compared to explicit neural networks, due to the need to find $\arg\min_{\boldsymbol{a}} E_\theta$. The value of $\boldsymbol{a}$ that satisfies this minimum condition cannot be found directly, and instead must be found using some type of iterative optimization process. PECM utilizes a Markov chain Monte Carlo (MCMC) optimization method known as stochastic gradient Langevin dynamics (SGLD) [12, 27]. Given some training sample index i, some counter-example index n, and some markov chain iteration index k, the formula for SGLD is:

$${}^{k}\tilde{\boldsymbol{a}}_i^n = {}^{k-1}\tilde{\boldsymbol{a}}_i^n - \frac{\lambda}{2}\nabla_{\boldsymbol{a}} E_\theta(\boldsymbol{o}_i, {}^{k-1}\tilde{\boldsymbol{a}}_i^n) + \lambda\omega^k, \omega^k \, \mathcal{N}(0, \sigma)$$

For each counter-example, the MCMC chain is run a fixed number of langevin iterations, defined by the parameter K. To increase the stability of the training process, a gradient penalty is added [12], defined as:

$$\mathcal{L}_{\text{grad}} = \sum_{i=1}^{I} \sum_{n=1}^{N-1} \max\left(0, ||\nabla_{\boldsymbol{a}} E_\theta(\boldsymbol{o}_i, {}^{K}\tilde{\boldsymbol{a}}_i^n)||_\infty - M\right)^2$$

M is a scaling factor known as the gradient margin that determines the relative magnitude of the gradient vs the noise in SGLD. To combine the InfoNCE loss and gradient penalty into the overall PECM loss function, the two losses are added together as $\mathcal{L} = \mathcal{L}_{\text{InfoNCE}} + \mathcal{L}_{\text{grad}}$ [12].

3.1 Theoretical Advantages

Energy-based models have a number of theoretical advantages which make them ideal for robotics and human pose mapping applications. One such advantage is in modeling behaviors with discontinuities [12]. Explicit neural network models are comprised of repeated layers of continuous functions such as addition, multiplication, and some non-linear activation function. All of these are continuous functions, and no combination or nesting of continuous functions can produce a nonlinear function. Therefore, any time an explicit neural network model is tasked with learning discontinuous behaviors, the resulting model will represent, at best, a very steep continuous slope connecting the two sides of the discontinuity. By contrast, EBMs model discontinuities natively, with arbitrarily high precision. Because EBM inference is performed with a gradient descent step that finds the local minimum energy value for a given observation, any observation that is in the "watershed" of a certain minimum energy action will predict that action. Thus, infinitesimal changes to an EBM's input observation can result in arbitrarily large changes in the model's predicted action output, as would be expected when modeling a discontinuity.

One case where effective modeling of discontinuities is helpful is in handling robot manipulator contact dynamics, where the system behavior changes completely once the manipulator has made contact with a static object [13]. Another case where modeling discontinuities is useful is in robotics path planning problems where a robot must swerve either right or left to avoid an obstacle, but selecting the midpoint of those paths would result in the robot crashing into the obstacle. In general, any problem which requires an agent to select between more than one distinct mode of operation contains a discontinuity. In the particular case of PECM, there are often cases where multiple different robot arm actions could correspond to a given human pose observation, but the average of all or any of them would not. Therefore, PECM must select between multiple discontinuous behavior options.

Another context where EBMs have theoretically advantageous qualities is when extrapolating to data outside the training set. in [12], multiple test tasks were performed where an

EBM and an explicit network were trained on a small amount of data, and the generalization performance of those networks were compared. In one benchmark task, the networks were trained with data from a variety of one dimensional functions with discontinuities. In each case, the EBM was able to correctly infer the function's behavior outside the training set by linearly extrapolating the function's behavior along the edges of the training distribution. The explicit network was less successful in doing this, resulting in visible irregularities around the training data boundary. In [27], EBMs were compared against two other generative models across five different out-of-distribution datasets. In this testing, EBMs perfomed the best overall, and had the best generalizaton in four of the five tasks. These results indicate that EBMs are good at finding the underlying patterns in data, even when tested on samples dissimilar to what they were trained on. This is useful for PECM, where the goal is to learn a mapping from any arbitrary human arm pose to a corresponding robot manipulator pose, but the dataset will not include samples of every possible pose.

Another benchmark task that was performed in [12] was visual coordinate regression task, where an EBM and an explicit network were presented with an image containing a single green dot. The goal of the task was for each network model to determine the coordinate position of the dot. The EBM performed well in this task, correctly generalizing to out-of-distribution data with only 10 training samples. The explicit network, by contrast, was unable to generalize outside of the training set even when given 30 training samples. The fact that EBMs are unusually good at extrapolation and generalization makes them ideal for tasks where only a small dataset is available, such as in PECM.

3.2 Model Training and Testing

3.2.1 Data Collection

The goal of the PECM pose mapping model is to solve the correspondence problem, thus enabling any human pose to be mapped to some corresponding robot pose. In order to train

the PECM model, a dataset of human motions and corresponding robot joint trajectories was therefore collected. To ensure that the quality of the motions and corresponding robot trajectories, all motions in the dataset were designed and recorded by trained dancers. Since dancers have an excellent understanding of human body dynamics and are trained to think flexibly about how motions can be represented in different contexts, it made sense that they would be uniquely qualified to provide the motions for the dataset.

The training dataset consisted of 11 different corresponding human/robot motion pairs, with 6 sample demonstrations for each. Each recorded motion was roughly 5 seconds long, for a total of approximately 5.5 minutes of dataset recordings. While the dataset here was small, one of the research questions here was to see if a small quantity of diverse pose demonstrations would be sufficient for a neural network model to effectively generalize to a much wider range of pose correspondences.

For each human motion in the dataset, one human pose trajectory was recorded using an OAK-D depth camera as shown in Figure 3.1. The OAK-D camera was loaded with the BlazePose human pose estimation model proposed in [28]. This enabled 3D full-body human pose tracking, where keypoints along the body were quickly and accurately captured. During each pose recording, human subjects were instructed to only move their arms when generating each motion, as the focus of this research was to try to find a correspondence between human arms motions and robot arm motions. Finding a correspondence between an entire human body and a robot was beyond the scope of this thesis, and needs further study.

For each human pose trajectory, one corresponding robot joint trajectory was also captured, such as the joint trajectory performed in Figure 3.2. The robot used for the data collection and testing process is a Trossen Robotics LoCoBot WX200 Rover, with a 5 degree-of-freedom (DoF) manipulator. This robot manipulator was kinesthetically moved by the same human subject who recorded the human pose trajectory. The human subject was instructed to move the manipulator in a way that they felt represented "the same motion" as

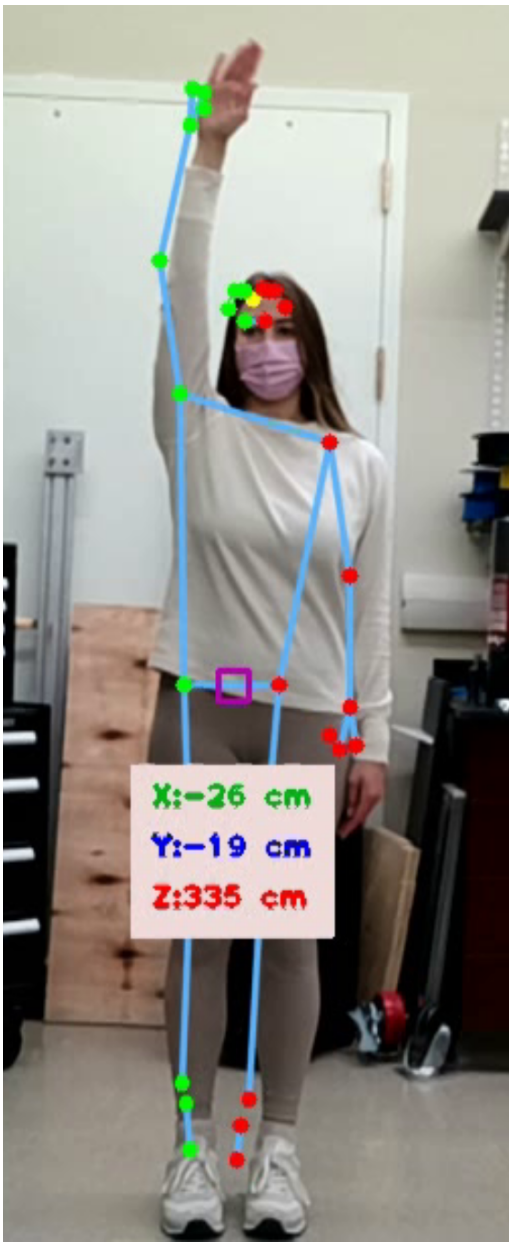


Figure 3.1: An example of the OAK-D depth camera recording a human motion demonstration

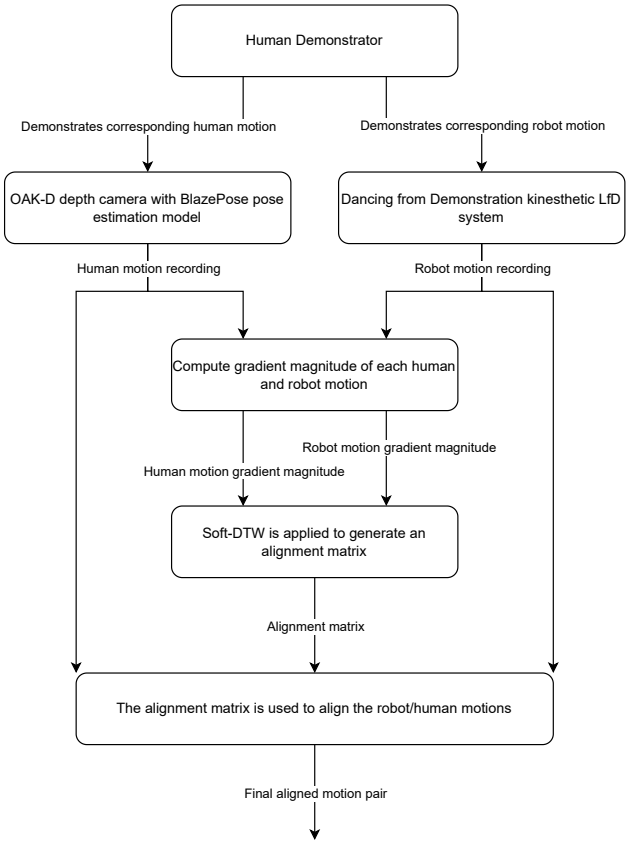
Figure 3.2: The robot playing back the corresponding human demonstrated robot motion

the motion they had performed during the human pose recording session. The manipulator joint trajectories were then recorded at 10Hz using the angle sensors built into the servo motors. This robot motion recording process was done using the kinesthetic LfD technique developed for DfD.

Once all the human and robot motions were collected, the data needed to be time-

aligned before it could be used for training. It seems reasonable to assume that a movement performed by the human subject should always correspond to a movement of the robot manipulator. Therefore, the gradient magnitude of each corresponding pair of human and robot trajectories should be roughly the same. Thus, time-alignment was carried out by performing Soft-DTW [23] on the gradient magnitude of each pair of motions. Soft-DTW was used in place of DTW [21] because of its greater tolerance to noise and variation between the two signals to be aligned. The full data collection and alignment procedure can be found in Figure 3.3.

Figure 3.3: The full PECM data collection process



3.2.2 Training

Two neural network models were trained with this dataset. The first is the PECM model, and the second is an explicit neural network baseline model using a mean squared error (MSE)

loss function. Both models use a multilayer perceptron (MLP) structure. Hyperparameters for each model can be found in Table 3.1 and Table 3.2.

Table 3.1: Pose Energy Correspondence Mapping Model Hyperparameters

Hyperparameter	**Value**
Training iterations	100000
Batch size	512
Sequence length	3
Learning rate	0.001
Learning rate decay	0.99
Learning rate decay steps	100
Network width	256
Network depth	8
Activation function	ReLU
Layer spectral norm	Not used
Training counter-examples	8
Action boundary buffer	0.05
Gradient penalty	Final step only
Gradient margin	1
Langevin iterations	100
Langevin learning rate init.	0.1
Langevin learning rate final	0.00001
Langevin polynomial decay power	2
Langevin delta action clip	0.1
Langevin noise scale	1
Langevin 2nd iteration learning rate	Not used

Most of these hyperparameters were selected because they were found to produce the best results in benchmarks from [12]. Additionally, some parameters, such as model size, training iterations, or batch size, were set to the same value in each model to ensure a "fair" comparison between PECM and the baseline technique. However, the sequence length of each of these models was set to 3, unlike any of the configurations described for those benchmarks. This configuration choice was made in order to better enable the network to learn to move like a human. More specifically, human muscles work by applying a force to tendons and ligaments, which pull on bones, resulting in body motion. According to newton's second law, $F = ma$, and the mass of human body parts as well as robot joints remains constant throughout any given motion. Therefore, throughout any human motion, muscular

Table 3.2: Mean Squared Error Baseline Hyperparameters

Hyperparameter	**Value**
Training iterations	100000
Batch size	512
Sequence length	3
Learning rate	0.0005
Learning rate decay	0.99
Learning rate decay steps	100
Network width	256
Network depth	8
Activation function	ReLU
Dropout rate	0.1

force applied is proportional to the acceleration of the body part that muscle connects to. This means that in order for a neural network to properly learn the ways in which humans apply force using our muscles, the network training data must include information about the acceleration of the body parts connected to those muscles. Using numerical differentiation, it is possible to compute the acceleration of some object given three measurements of the position of that object at different moments in time. Therefore, a sequence length of 3 provides the neural network with all the information it needs to accurately approximate the forces being exerted by a human's muscles throughout a given motion in the dataset. Having this information should allow the network to more effectively learn how to map motions from the human demonstrator to the robot.

3.2.3 Testing

In order to test the trained models, a set of three test motions was collected using the same procedure that was described in Section 3.2.1. Given each of these human test motions as input, inference was performed with each model to generate a corresponding robot motion. These motions as well at the human demonstrated robot motions in the test set were then played back on the robot using DfD and a video of each motion was recorded. A survey of these motion videos was then created, where survey participants were shown video of each human motion and a corresponding robot motion, and asked to rate these robot motions on

a variety of criteria. The order of questions as well as the order of sections in the survey was randomized, so that survey participants did not know which motions were which.

All survey questions involved survey participants rating the robots motion performance on a scale of 1 (the worst) to 5 (the best). There were six categories of questions that survey participants answered. The first two categories attempted to measure the quality of the correspondence between each human motion and robot motion. The first of these correspondence questions asked specifically about the quality of the movement correspondence, whereas the second correspondence question asked about the degree of correspondence of the emotional feeling of each motion. Each of these question explicitly asked survey participants how well the motions corresponded with each other. The next four categories of questions were all asked in the form of a Likert scale, where survey participants were asked to rate the robot motion on a scale according to some criteria or attribute of the motion. Question categories three and four were each taken from the Godspeed Questionnaire [29], which is a standardized set of survey questions for evaluating human-robot interaction techniques. The Godspeed Questionnaire itself contains five sections, but only two of them were relevant to evaluating PECM. The third question category was the Godspeed anthropomorphism questions, in order to evaluate how humanlike the generated motions appeared. Likewise, the fourth question category was the Godspeed animacy questions, in order to evaluate how lifelike the generated motions appeared. The fifth and sixth question categories each consisted of a single question, chosen to fill in the gaps and evaluate each motion correspondence in a way that none of the preceding questions did. The fifth question category asked for a rating of the smoothness of the robot motion, and the sixth asked about the balance and evenness of the robot motion.

Much like the human demonstrators who provided all of the motions for the data collection process, all four of the survey participants were experienced dancers and choreographers, with the least experienced among them having over 15 years of dance experience and more than 5 years of choreography experience. The purpose of having dancers provide the sur-

vey responses was to ensure that the motions would be evaluated by someone with a deep understanding of body dynamics and motion, theoretically making their responses more accurate.

Figure 3.4: Average benchmark results from each question category. See Appendix A for full results of each individual category.

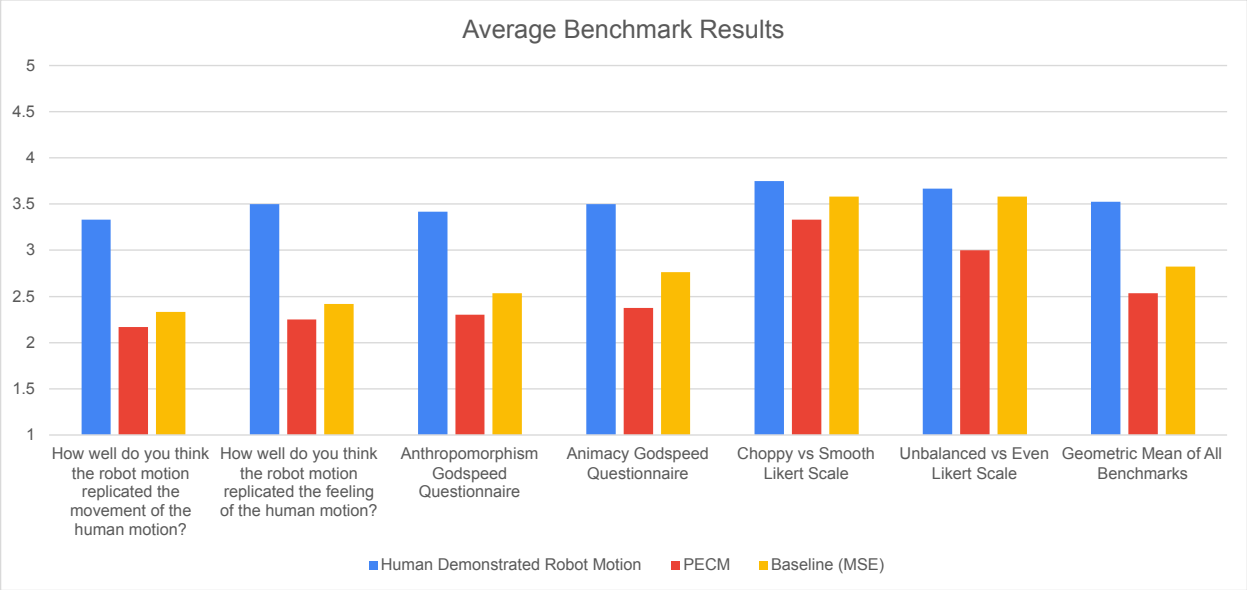


Table 3.3: Average benchmark results from each question category. See Appendix A for full results of each individual category.

Survey Question Category	**Human Demonstrated Robot Motion**	**PECM**	**Baseline (MSE)**
How well do you think the robot motion replicated the movement of the human motion?	3.333333333	2.166666667	2.333333333
How well do you think the robot motion replicated the feeling of the human motion?	3.5	2.25	2.416666667
Anthropomorphism Godspeed Questionnaire	3.416666667	2.3	2.533333333
Animacy Godspeed Questionnaire	3.5	2.375	2.763888889
Choppy vs Smooth Likert Scale	3.75	3.333333333	3.583333333
Unbalanced vs Even Likert Scale	3.666666667	3	3.583333333
Geometric Mean of All Benchmarks	3.524953215	2.536458017	2.823775369

As shown in Figure 3.4 and Table 3.3, PECM performed worse than the MSE baseline in every tested category. It should be noted that the hyperparameters of both models were not tuned, which likely contributed negatively to these results. While PECM performed

only slightly worse in correspondence metrics, it performed significantly worse in many of the more qualitative metrics including Anthropomorphism, Animacy, and evenness. This indicates that while the degree of correspondence was nearly as high for PECM as the baseline, the nature of the motions generated by pecm was significantly less humanlike and more robotic.

An important question to answer is, given all of the theoretical advantages that PECM has, why did it fail to produce good results in this use case? It seems likely that this failure is partially the result of a property of EBMs proven in [12], the fact that EBMs are capable of representing a larger class of functions than normal explicit networks such as the MSE baseline. While this would at first appear to be beneficial, this greater representational power does mean that EBMs with equal model size have a greater capacity for overfitting than explicit networks.

One factor that compounds this issue is the very small size of the training dataset used in this research, which is largely due to the time consuming nature of collecting motion correspondence data. In order to collect such data, a human subject must invent a new motion, record that motion in controlled conditions in view of a camera, invent a corresponding robot arm motion, and then use DfD to kinesthetically teach that motion to the robot. This process can take multiple minutes in some cases just to record one motion correspondence that is a few seconds long. It can therefore take many hours to collect a dataset of any decent size.

Additionally, not only is the dataset small, it also only contains only a small subset of all possible human-robot motions. When an EBM was chosen as the neural network technique underpinning PECM, it was hoped that the greater extrapolation capabilities of EBMs would enable the model to learn a generalized mapping of all possible human motions to all possible robot motions. Indeed, EBMs have been shown in [12] to perform well in cases where the training dataset is small, being randomly sampled from a much larger and more complete dataset. However, due to way the InfoNCE loss function works, any part

of the problem space that does not contain training samples will eventually be filled with counterexamples, and the EBM will learn to avoid predicting those values. Therefore, in cases where large regions of the problem space are missing from the training dataset entirely, EBMs will struggle. As a result, PECM generates motions which tend to be very "modal", switching rapidly to whatever region of the problem space contained the most similar motion from the training set.

Subjectively, this issue could be observed in many of the motions that were generated by PECM. The motions appeared very jerky and aggressive, the robot manipulator often moving rapidly even when the human demonstrator had only moved a small amount. These moments of jerkiness were interspersed with periods of calm, where the PECM motions were very smooth and high-quality. It seems likely that these regions of high quality results were the areas of the problem space that contained the most training data. By contrast, the MSE baseline generated motions that were never very jerky, but it never generated any portions of motions which reached the same level of quality as PECM sometimes did. Additionally, the MSE generated motions were always subdued, as if the network never quite knew what to do. Thus, it would seem that EBMs degrade differently than MSE networks with insufficient training data. Explicit networks degrade by becoming less and less "confident" in their predictions, resulting in motions that are dull and full of hesitation. By contrast, EBMs degrade by remaining decisive in their predictions, but sometimes predicting the incorrect thing. Unfortunately, in the case of solving the correspondence problem, generating a hesitant motion is better than generating a confidently wrong motion.

Chapter 4

Conclusion

4.1 Conclusion

In this thesis, two robot control techniques were presented: DfD, a kinestethic LfD framework, and PECM, which is an observational LfD system conceptualized as a replacement for the motion recording portion of DfD. DfD was developed as a simple and efficient motion learning and playback system, capable of motion modulation with real-time performance. This modulation process could be used either for motion smoothing, or for creative motion generation. DfD achieved its goals, being used successfully to enable a robot-human duet dance performance. PECM was then designed as an extension of DfD, making use of an Energy-based model to solve the correspondence problem and allow new motions to be demonstrated much more easily. Robot motions generated by PECM were then compared to motions generated by a baseline in a survey consisting of six different qustion categories. PECM performed worse than the baseline in every category of the survey. The reasons for this failure appear to be a combination of unforeseen deficiencies in the generalization behavior of Energy-based models, as well as a lack of training data resulting in overfitting.

4.2 Future Work

One area of future research related to this thesis is clear: develop a functional version of PECM. There are two main categories of approaches by which this could be done. One possibility would be to continue developing a machine learning approach, potentially involving some other neural network variety besides energy based models. For this to be practical, it seems likely necessary to acquire significantly more training data than what was used in this thesis, in order to avoid the same problems that PECM encountered. In fact, it is possible that the collection of more training data alone would be enough to make PECM work, even without any other technical changes. However, collecting that much training data would be a slow process. Related to this, another interesting research thread would be to investigate if there is any way to collect training data for this task more quickly and easily. If such a method could be developed, then solving the correspondence problem with machine learning would become significantly easier.

The other general approach to potentially solve the correspondence problem would be to place less emphasis on machine learning, and instead make use of inverse and forward kinematics based techniques. Given that humans and robots are both made of some set of rigid links connected by joints, robot kinematics techniques could in theory be applied to humans as well. It therefore seems possible to develop some method that can map poses from a human to a robot, entirely without machine learning. A hybrid of these approaches could also be developed. One idea in this area would be to use robot kinematics to develop some sort of model that could interface with a neural network in order to make its learning task easier.

Unrelated to PECM, research could also be done to expand and design further capabilities for DfD. The motion modulation approach that DfD uses to generate novel motions is in a relatively unexplored area of research, and it could potentially be extended with new capabilities for dynamic motion generation. Alternatively, the motion smoothing capabilities also provided by DfD could potentially be improved as well.

Appendix A

Extra Figures and Tables

Figure A.1: Full results for the movement correspondence survey category.

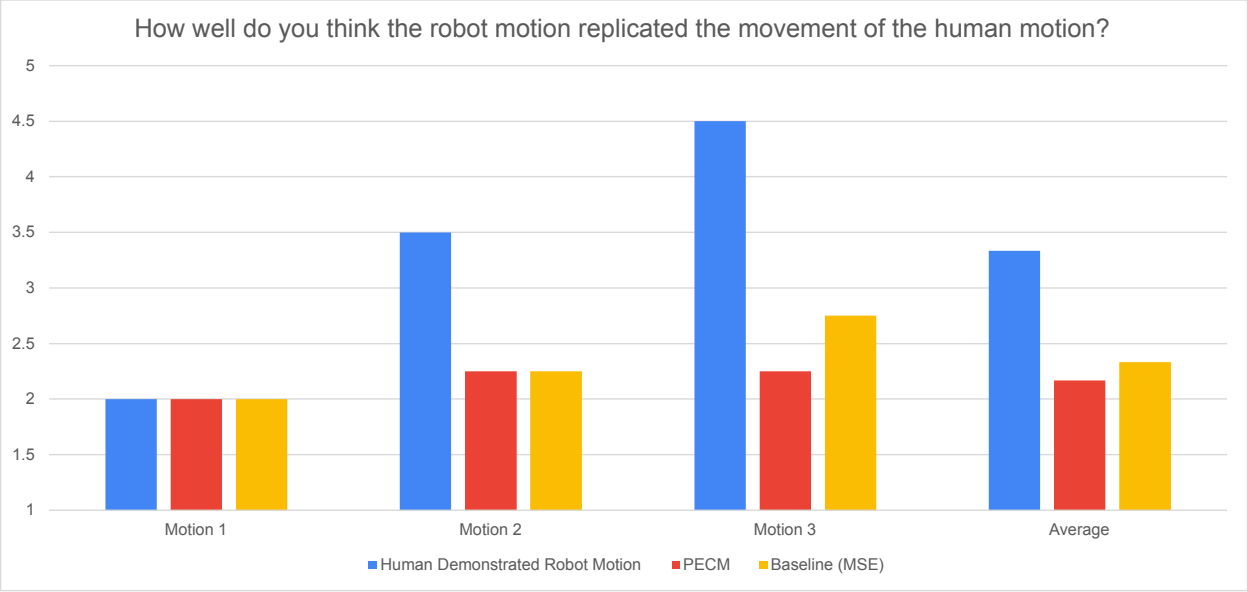


Table A.1: Full results for the movement correspondence survey category.

How well do you think the robot motion replicated the movement of the human motion?	**Human Demonstrated Robot Motion**	**PECM**	**Baseline (MSE)**
Motion 1	2	2	2
Motion 2	3.5	2.25	2.25
Motion 3	4.5	2.25	2.75
Average	3.333333333	2.166666667	2.333333333

Figure A.2: Full results for the feeling correspondence survey category.

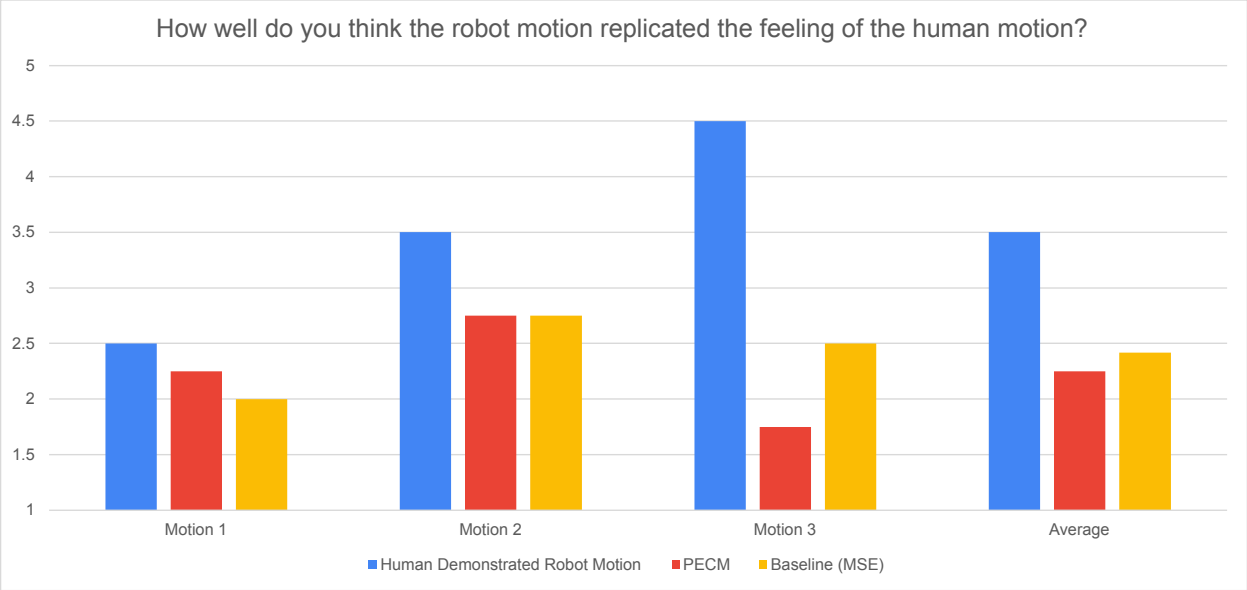


Table A.2: Full results for the feeling correspondence survey category.

How well do you think the robot motion replicated the feeling of the human motion?	**Human Demonstrated Robot Motion**	**PECM**	**Baseline (MSE)**
Motion 1	2.5	2.25	2
Motion 2	3.5	2.75	2.75
Motion 3	4.5	1.75	2.5
Average	3.5	2.25	2.416666667

Figure A.3: Full results for the anthropomorphism survey category.

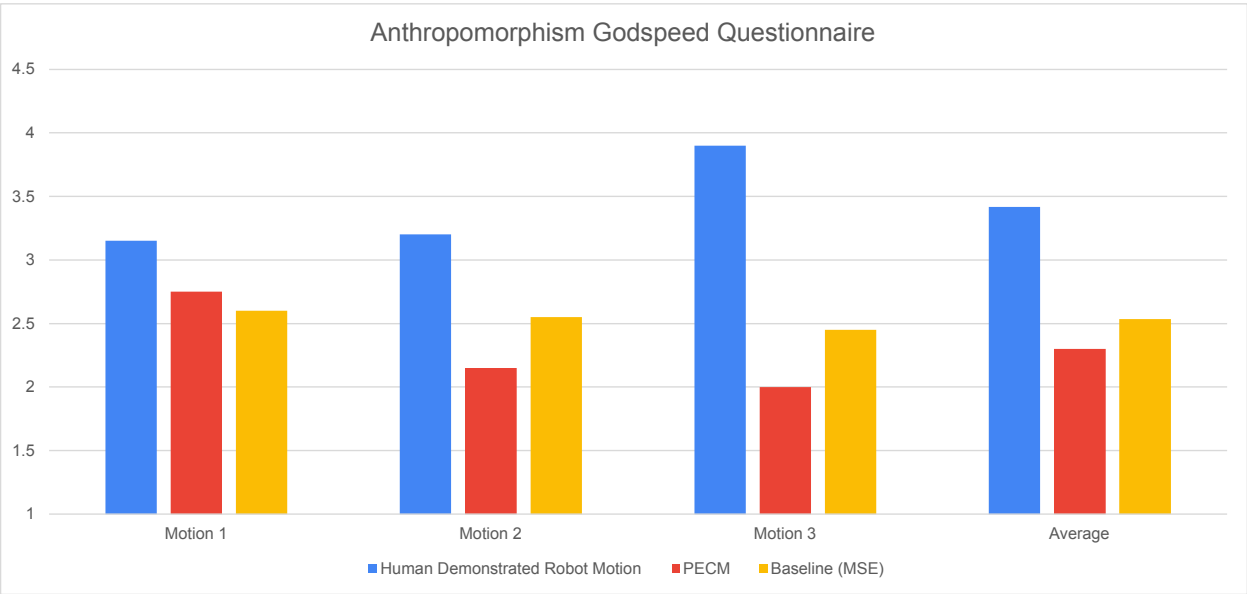


Table A.3: Full results for the anthropomorphism survey category.

Anthropomorphism Godspeed Questionnaire	Human Demonstrated Robot Motion	PECM	Baseline (MSE)
Motion 1	3.15	2.75	2.6
Motion 2	3.2	2.15	2.55
Motion 3	3.9	2	2.45
Average	3.416666667	2.3	2.533333333

Figure A.4: Full results for the animacy survey category.

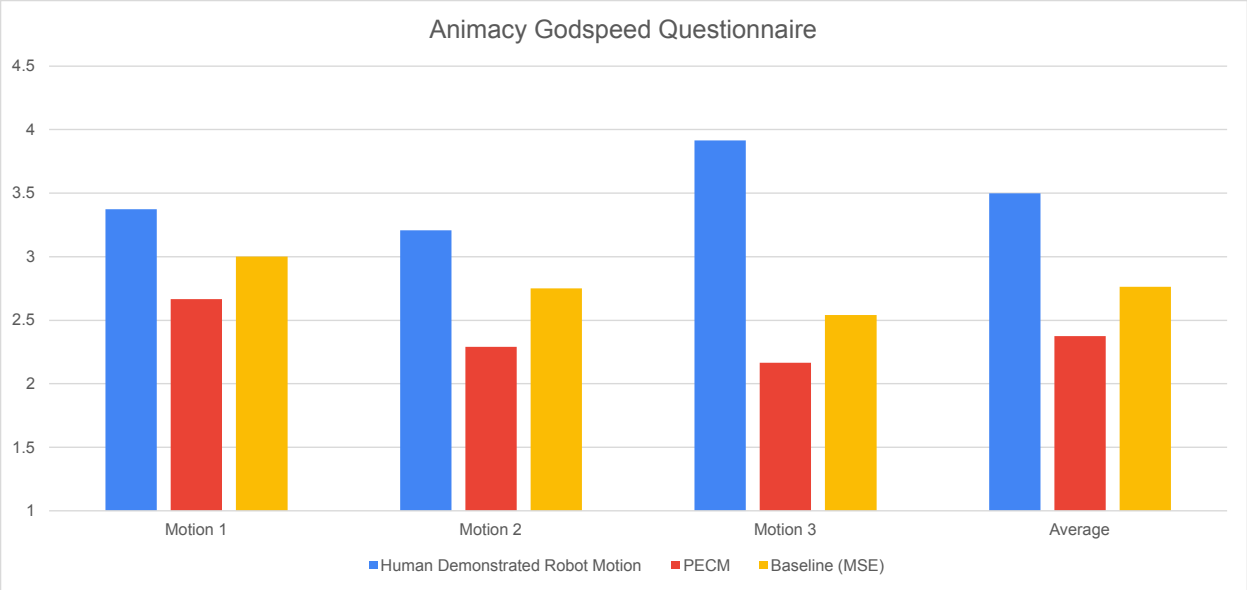


Table A.4: Full results for the animacy survey category.

Animacy Godspeed Questionnaire	Human Demonstrated Robot Motion	PECM	Baseline (MSE)
Motion 1	3.375	2.666666667	3
Motion 2	3.208333333	2.291666667	2.75
Motion 3	3.916666667	2.166666667	2.541666667
Average	3.5	2.375	2.763888889

Table A.5: Full results for the choppy vs smooth survey category.

Choppy vs Smooth Likert Scale	Human Demonstrated Robot Motion	PECM	Baseline (MSE)
Motion 1	3.25	3.5	3.75
Motion 2	3.5	3.5	3.25
Motion 3	4.5	3	3.75
Average	3.75	3.333333333	3.583333333

Figure A.5: Full results for the choppy vs smooth survey category.

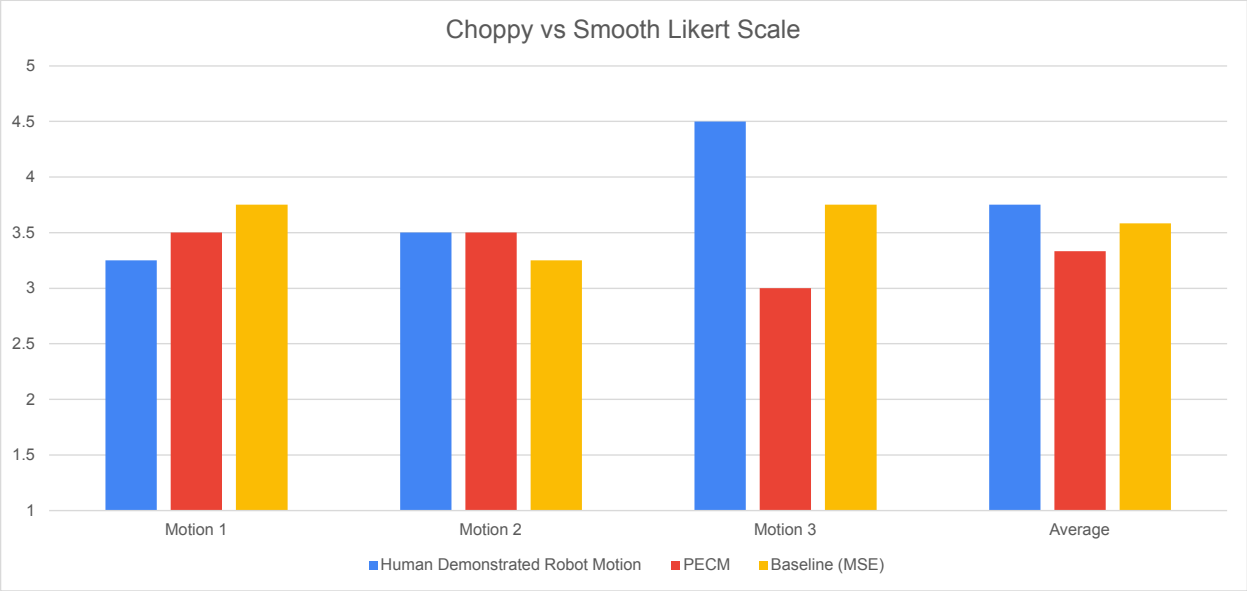


Figure A.6: Full results for the unbalanced vs even survey category.

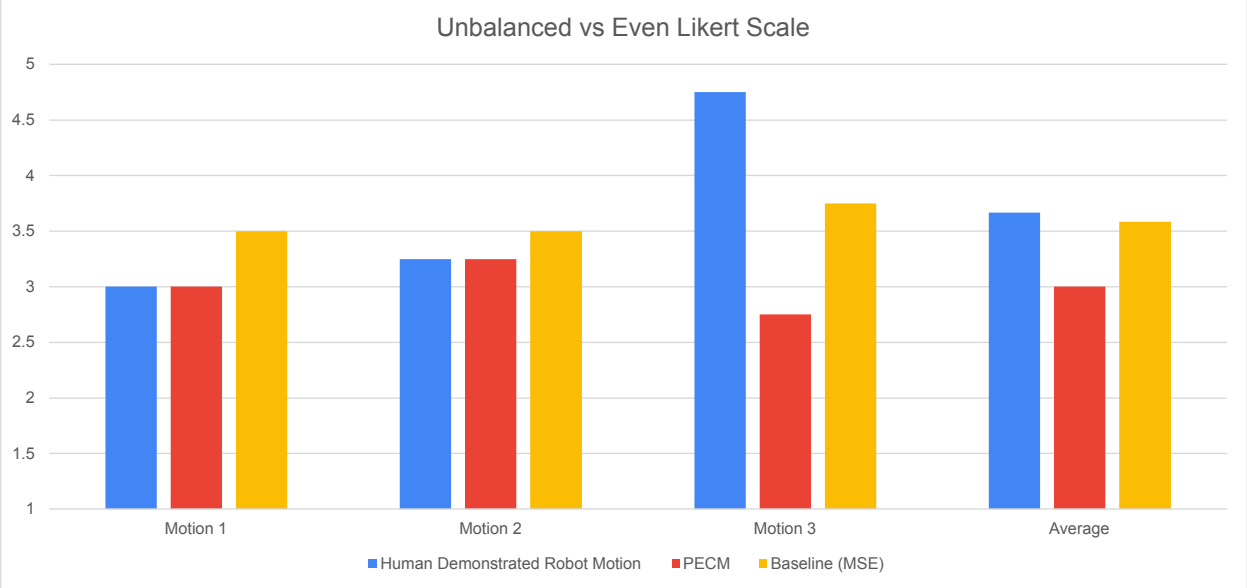


Table A.6: Full results for the unbalanced vs even survey category.

Unbalanced vs Even Likert Scale	**Human Demonstrated Robot Motion**	**PECM**	**Baseline (MSE)**
Motion 1	3	3	3.5
Motion 2	3.25	3.25	3.5
Motion 3	4.75	2.75	3.75
Average	3.666666667	3	3.583333333

Vita

Charles Dietzel was born on June 14, 1999 in Fairfax County, Virginia. He graduated from Chantilly High School in Chantilly, Virginia in 2017. He received his Bachelor of Science in Electrical Engineering from Virginia Commonwealth University, in Richmond, Virginia in 2021. He has worked for the past year and a half as a graduate researcher in the HIVE Lab at Virginia Commonwealth University.

References

[1] Z. Zhu and H. Hu, "Robot Learning from Demonstration in Robotic Assembly: A Survey," *Robotics*, vol. 7, no. 2, p. 17, Jun. 2018, number: 2 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2218-6581/7/2/17

[2] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation Learning: A Survey of Learning Methods," *ACM Computing Surveys*, vol. 50, no. 2, pp. 21:1–21:35, Apr. 2017. [Online]. Available: https://doi.org/10.1145/3054912

[3] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent Advances in Robot Learning from Demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 297–330, 2020, _eprint: https://doi.org/10.1146/annurev-control-100819-063206. [Online]. Available: https://doi.org/10.1146/annurev-control-100819-063206

[4] M. Schneider and W. Ertel, "Robot Learning by Demonstration with local Gaussian process regression," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2010, pp. 255–260, iSSN: 2153-0866.

[5] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, May 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889008001772

[6] P. Martin, K. Sicchio, C. Dietzel, and A. Olivo, "Towards A Framework For Dancing Beyond Demonstration," in *Proceedings of the 8th International Conference on Movement and Computing*, ser. MOCO '22. New York, NY, USA: Association for Computing Machinery, Jun. 2022, pp. 1–7. [Online]. Available: https://doi.org/10.1145/3537972.3537981

[7] A. Kim, H. Kum, O. Roh, S. You, and S. Lee, "Robot gesture and user acceptance of information in human-robot interaction," in *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, Mar. 2012, pp. 279–280, iSSN: 2167-2148.

[8] J. de Wit, B. Willemsen, M. de Haas, E. Krahmer, P. Vogt, M. Merckens, R. Oostdijk, C. Savelberg, S. Verdult, and P. Wolfert, "Playing Charades with a Robot: Collecting a Large Dataset of Human Gestures Through HRI," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, Mar. 2019, pp. 634–635, iSSN: 2167-2148.

[9] R. Saunders and P. Gemeinboeck, "Performative Body Mapping for Designing Expressive Robots," in *Proceedings of the 9th International Conference on Computational Creativity (ICCC 2018)*, Jun. 2018, pp. 280–287.

[10] J. Ho and S. Ermon, "Generative Adversarial Imitation Learning," *arXiv:1606.03476 [cs]*, Jun. 2016, arXiv: 1606.03476. [Online]. Available: http://arxiv.org/abs/1606.03476

[11] J. Fu, K. Luo, and S. Levine, "Learning Robust Rewards with Adversarial Inverse Reinforcement Learning," *arXiv:1710.11248 [cs]*, Aug. 2018, arXiv: 1710.11248. [Online]. Available: http://arxiv.org/abs/1710.11248

[12] P. Florence, C. Lynch, A. Zeng, O. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, "Implicit Behavioral Cloning," *arXiv:2109.00137 [cs]*, Aug. 2021, arXiv: 2109.00137. [Online]. Available: http://arxiv.org/abs/2109.00137

[13] B. Bianchini, M. Halm, N. Matni, and M. Posa, "Generalization Bounded Implicit Learning of Nearly Discontinuous Functions," *arXiv:2112.06881 [cs]*, Apr. 2022, arXiv: 2112.06881. [Online]. Available: http://arxiv.org/abs/2112.06881

[14] R. F. Prudencio, M. R. O. A. Maximo, and E. L. Colombini, "A Survey on Offline Reinforcement Learning: Taxonomy, Review, and Open Problems," *arXiv:2203.01387 [cs, stat]*, Mar. 2022, arXiv: 2203.01387. [Online]. Available: http://arxiv.org/abs/2203.01387

[15] R. Agarwal, D. Schuurmans, and M. Norouzi, "An Optimistic Perspective on Offline Reinforcement Learning," *arXiv:1907.04543 [cs, stat]*, Jun. 2020, arXiv: 1907.04543. [Online]. Available: http://arxiv.org/abs/1907.04543

[16] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative Q-Learning for Offline Reinforcement Learning," *arXiv:2006.04779 [cs, stat]*, Aug. 2020, arXiv: 2006.04779. [Online]. Available: http://arxiv.org/abs/2006.04779

[17] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4RL: Datasets for Deep Data-Driven Reinforcement Learning," *arXiv:2004.07219 [cs, stat]*, Feb. 2021, arXiv: 2004.07219. [Online]. Available: http://arxiv.org/abs/2004.07219

[18] I. Kostrikov, A. Nair, and S. Levine, "Offline Reinforcement Learning with Implicit Q-Learning," *arXiv:2110.06169 [cs]*, Oct. 2021, arXiv: 2110.06169. [Online]. Available: http://arxiv.org/abs/2110.06169

[19] W. Zhou, S. Bajracharya, and D. Held, "PLAS: Latent Action Space for Offline Reinforcement Learning," *arXiv:2011.07213 [cs]*, Nov. 2020, arXiv: 2011.07213. [Online]. Available: http://arxiv.org/abs/2011.07213

[20] B. Reiner, W. Ertel, H. Posenauer, and M. Schneider, "LAT: A simple Learning from Demonstration method," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2014, pp. 4436–4441, iSSN: 2153-0866.

[21] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, Feb. 1978, conference Name: IEEE Transactions on Acoustics, Speech, and Signal Processing.

[22] F. Petitjean, A. Ketterlin, and P. Gançarski, "A global averaging method for dynamic time warping, with applications to clustering," *Pattern Recognition*, vol. 44, no. 3, pp. 678–693, Mar. 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S003132031000453X

[23] M. Cuturi and M. Blondel, "Soft-DTW: a Differentiable Loss Function for Time-Series," Feb. 2018, arXiv:1703.01541 [stat]. [Online]. Available: http://arxiv.org/abs/1703.01541

[24] P. Schillinger, S. Kohlbrecher, and O. von Stryk, "Human-robot collaborative high-level control with application to rescue robotics," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 2796–2802.

[25] P. H. Le-Khac, G. Healy, and A. F. Smeaton, "Contrastive Representation Learning: A Framework and Review," *IEEE Access*, vol. 8, pp. 193 907–193 934, 2020, conference Name: IEEE Access.

[26] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation Learning with Contrastive Predictive Coding," Jan. 2019, arXiv:1807.03748 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1807.03748

[27] Y. Du and I. Mordatch, "Implicit Generation and Generalization in Energy-Based Models," *arXiv:1903.08689 [cs, stat]*, Jun. 2020, arXiv: 1903.08689. [Online]. Available: http://arxiv.org/abs/1903.08689

[28] V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, and M. Grundmann, "BlazePose: On-device Real-time Body Pose tracking," Jun. 2020, arXiv:2006.10204 [cs]. [Online]. Available: http://arxiv.org/abs/2006.10204

[29] C. Bartneck, D. Kulić, E. Croft, and S. Zoghbi, "Measurement Instruments for the Anthropomorphism, Animacy, Likeability, Perceived Intelligence, and Perceived Safety of Robots," *International Journal of Social Robotics*, vol. 1, no. 1, pp. 71–81, Jan. 2009. [Online]. Available: https://doi.org/10.1007/s12369-008-0001-3